How Do I Use the New Sandy Bridge Nodes?

June 21 & 27, 2012

NASA Advanced Supercomputing Division

Outline

- Pleiades Augmentation/Expansion
  - Removal of Some Harpertown and Addition of Sandy Bridge
- Main Differences among the Four Pleiades Processor Types
  - At the Node Level
  - Inside the Sandy Bridge Processor Cores
    - * Advanced Vector Extensions (AVX) and Floating Point Operations
- What Do I Need to Do to Run on the Sandy Bridge?
  - Do I Need to Recompile My Code?
  - Performance Comparisons of Some Applications (w. different compiler flags)
  - Do I Need to Change My PBS Script?
- Should I Run on the Sandy Bridge?
  - Availability Consideration
  - Memory Consideration
  - Performance and SBU Rates Consideration

Pleiades Augmentation/Expansion

	Harpertown	Nehalem	Westmere	Sandy Bridge
# of Racks	~~91~~ 64	20	74	24
# of Nodes	~~5,824~~ 4,096	1,280	4,672	1,728
# of Cores	~~46,592~~ 32,768	10,240	56,064	27,648
Peak TFlops	~~559~~ 393	120	658	575

- Total hardware: 182 racks, 11,776 nodes, 126,720 cores
- Total theoretical peak performance: 1.75 Pflops/s

Question? Use the Webex chat facility to ask the Host

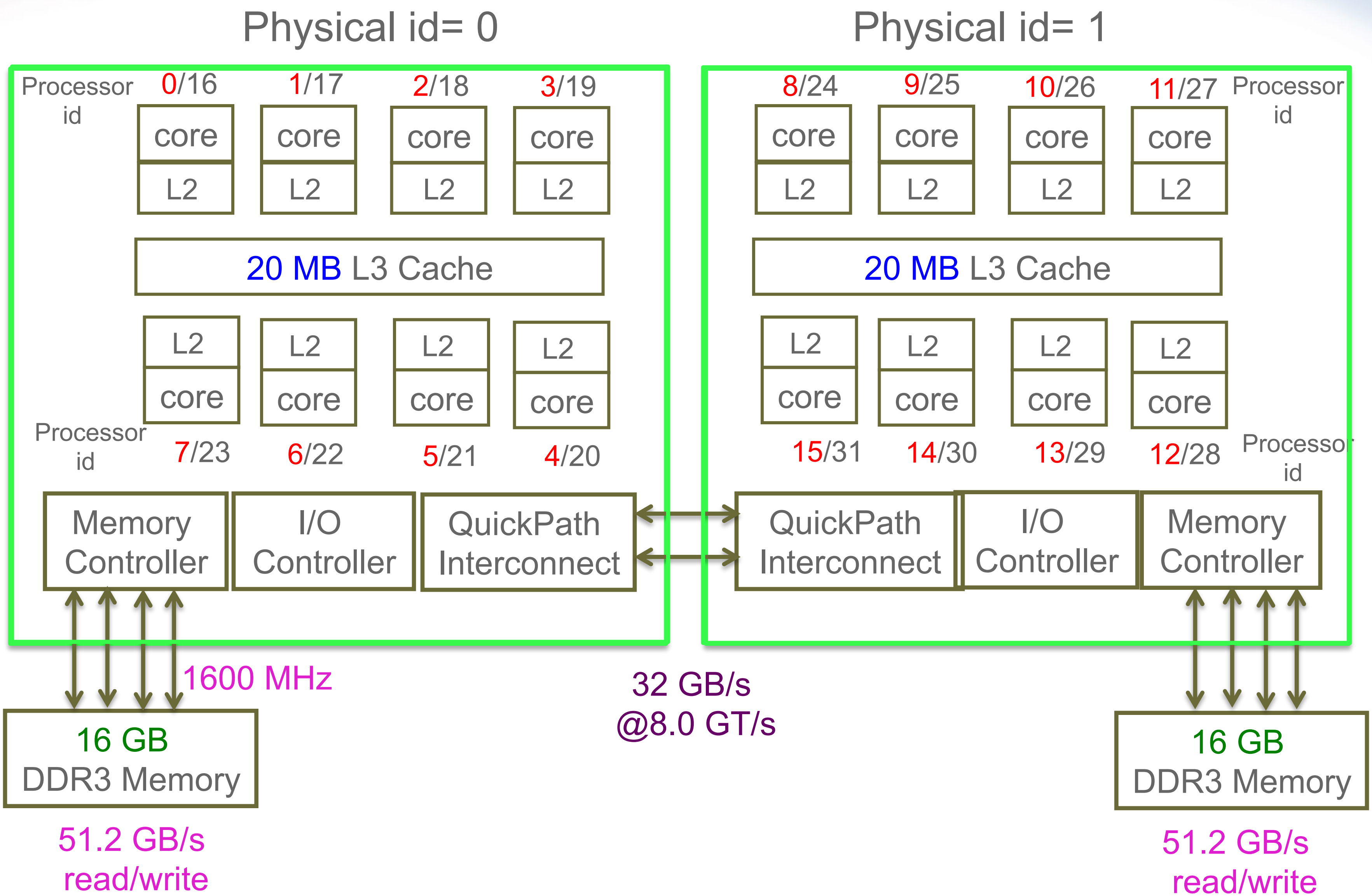
Main Differences among Pleiades Processors

	Harpertown	Nehalem	Westmere	Sandy Bridge
Intel Model	E5472	X5570	X5670/X5675*	E5-2670
# of Cores per Node	2 x 4-core = 8	2 x 4-core =8	2 x 6-core= 12	2 x 8-core= 16
CPU-Clock	3.0 GHz	2.93 GHz	2.93/3.06 GHz	2.6 GHz
DP FPs* per cycle per core	4	4	4	8
Largest Cache	6 MB*/2 cores	8 MB/4 cores	12 MB/6 cores	20 MB/8 cores
Memory per node	8 GB*	24 GB	24 GB*	32 GB*
Memory per Core	1 GB	3 GB	2 GB	2 GB
Memory Bandwidth per socket	25.6 GB/s read 12.8 GB/s write	32 GB/s	32 GB/s	51.2 GB/s
QPI between sockets	Not applied	25.6 GB/s	25.6 GB/s	32 GB/s
IB between nodes & to Lustre	4x DDR (4 x 5 Gb/s)	4x DDR/QDR* (4 x 5 Gb/s)	4x QDR (4 x 10 Gb/s)	4x FDR (4 x 14 Gb/s)

- Westmere racks 221 & 222 use X5675; rack 219 also includes Nvidia GPUs
- DP FPs: Double Precision Floating Point Operations per second
- Largest cache in Harpertown is L2; the others are L3
- Bigmem nodes: Harpertown nodes in rack 32 have 16 GB/node; 17 Westmere nodes have 48 GB/node; 4 Westmere nodes have 96 GB/node; Some Sandy Bridge nodes may have more memory in the near future
- The Nehalem racks use DDR Host Channel Adapters (HCAs) and QDR switches

Question? Use the Webex chat facility to ask the Host

Sandy Bridge Node Configuration



Question? Use the Webex chat facility to ask the Host

Improvements of Sandy Bridge Node over Westmere Node (not considering the cores)

- Larger L3 cache (2.5MB/core vs 2.0MB/core)
- Higher memory speed and bandwidth

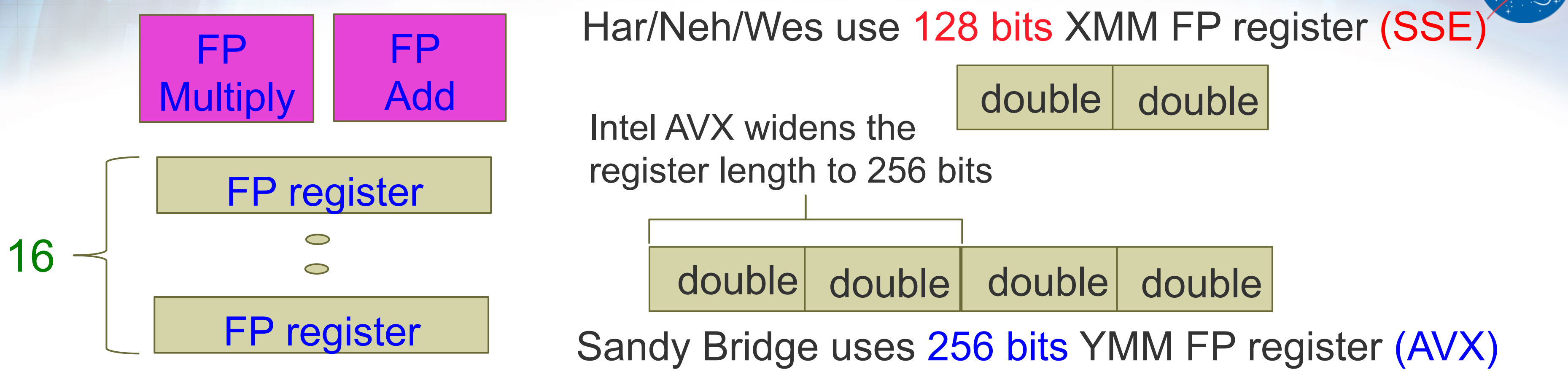
  (1600 MHz vs 1333 MHz; 4 channels vs 3 channels)
- Higher Quick Path Interconnect speed/bandwidth
- Faster communication between nodes via FDR vs QDR

An application can perform better due to these improvements alone.

Next, what has changed inside the core?

Question? Use the Webex chat facility to ask the Host

Floating Point Operations inside a Core



Sample instruction in a loop : X(i) = X(i) + Y(i)

Scalar Mode	Vectorization with SSE (Streaming SIMD Extensions)	Vectorization with AVX (Advanced Vector Extensions)
X	X1 | X0	X3 | X2 | X1 | X0
+	+	+
Y	Y1 | Y0	Y3 | Y2 | Y1 | Y0
=	=	=
X+Y	X1 + Y1 | X0 + Y0	X3+Y3 | X2+Y2 | X1+Y1 | X0+Y0

Vectorization of a loop = unrolling the loop so that it can take advantage of packed SIMD instructions to perform the same operation on multiple data in a single instruction

Question? Use the Webex chat facility to ask the Host

Floating Point Operations inside a Core (cont'd)

Number of SSE instructions: SSE - 70, SSE2 - 144, SSE3 - 13, SSSE3 - 16, SSE4.1 - 47, SSE4.2 – 7 (compiler flags: -xSSE4.1 or –xSSE4.2)

SSE includes instructions for arithmetic, logic, compare, convert, load/store, etc. operations, not all are related to floating point operations or use the XMM registers.

Number of AVX instructions: 12 (compiler flag: –xAVX)

- In each core, there are 16 floating point registers and 2 floating point functional units
- If a code is well vectorized with data pipelined in the 16 registers, and both functional units are busy, it can achieve
  - maximum 4 double precision Flops/cycle/core or 8 single precision Flops on Har/Neh/Wes with SSE
  - maximum 8 double precision Flops/cycle/core or 16 single precision Flops on Sandy Bridge with AVX

  Problem: many user codes do not achieve these maximum Flop rates
- If your code gets more Flops per cycle on Sandy Bridge than on the other three processor types, even with lower clock speeds (2.6 GHz vs 3.0/2.93 GHz), the time spent on floating point operations may be shorter

Question? Use the Webex chat facility to ask the Host

Taking Advantage of AVX

- Use compiler flag -vec-report2 (need –O2 and above) to get reports on why loops are not vectorized
- Modify source code to allow more vectorization

  (only inner loop can be vectorized; unless inlined, avoid function/subroutine calls in loops; no branches, number of loop iterations must be known; avoid non-unit stride or indirect addressing; no dependency; make code 32 bytes aligned, etc.)
- Use Intel MKL library (-mkl) which continues to be optimized for newer generations of Intel architecture (including AVX)
- References about vectorization and AVX
  - Requirements for Vectorizable Loops http://software.intel.com/en-us/articles/requirements-for-vectorizable-loops/
  - http://software.intel.com/en-us/avx/
  - Introduction to Intel Advanced Vector Extensions

    http://software.intel.com/en-us/articles/introduction-to-intel-advanced-vector-extensions/
  - Pdf file "Compiling for the Intel 2nd Generation Core processor family and the Intel AVX instruction set" http://software.intel.com/file/34217/
  - Practical Intel AVX Optimization on 2nd generation Intel Core Processors

    http://software.intel.com/en-us/articles/practical-intel-avx-optimization-on-2nd-generation-intel-core-processors/

Do I Need to Recompile My Code?

- No. If you do not care about performance, your existing executable that ran on Har/Neh/Wes should work on Sandy Bridge.
- Yes. If you want to explore getting better performances. We recommend:
  - Use latest Intel version 12 compiler on Pleiades
  - Experiment with –O2 vs –O3 and –ip (or –ipo if you have multiple source files)

    Note: Vectorization is enabled at –O2 and above; -O3 allows more loop optimization than –O2
  - Experiment with adding –xAVX (code runs on SAN only)
  - If you choose to use –xAVX, apply it to all source files

    (there is performance penalty with functions compiled with –xAVX, -xSSEn calling each other)
  - Experiment with –axAVX –xSSE4.1 (code runs on Har/Neh/Wes/San; gives both SSE and AVX code paths with SSE4.1 as the default; Which code path is used is determined at runtime)
- Why choosing Intel version 12 over version 11?
  - Version 11.1 accepts –xAVX (though not documented in man page) but performance may not be as good as version 12. Version 11.0 does not accept -xAVX
  - MKL 10.3 used in version 12 compilers includes more AVX optimization (dgemm/sgemm, all BLAS level 3 functions, …) than MKL 10.2 used in version 11 compilers
  - Even without –xAVX, version 12 may provide more optimization for your code
- Whatever combination you choose, verify correctness is important!!

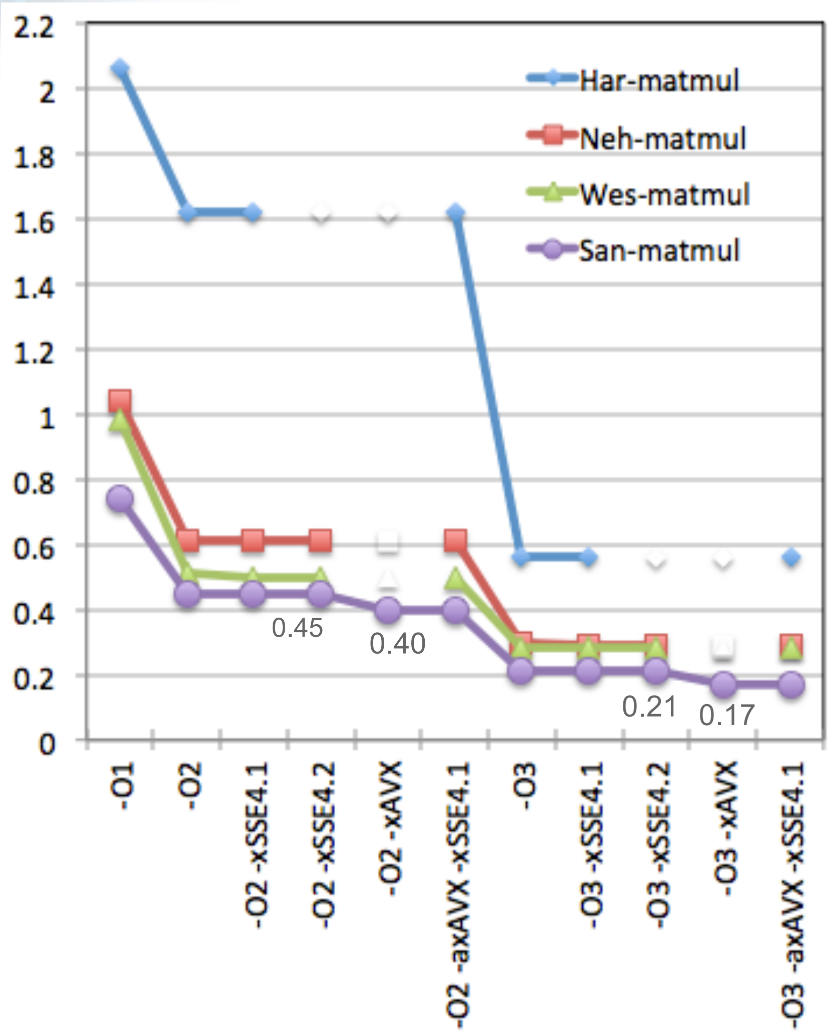
Effects of –x or –ax Choices at Run Time

Choice of –x or -ax	Harpertown	Nehalem	Westmere	Sandy Bridge
None (default to -mSSE2)	OK	OK	OK	OK
-xSSE4.1	OK	OK	OK	OK
-xSSE4.2	Abort	OK	OK	OK
-xAVX	Abort	Abort	Abort	OK
-axAVX –xSSE4.1	OK	OK	OK	OK

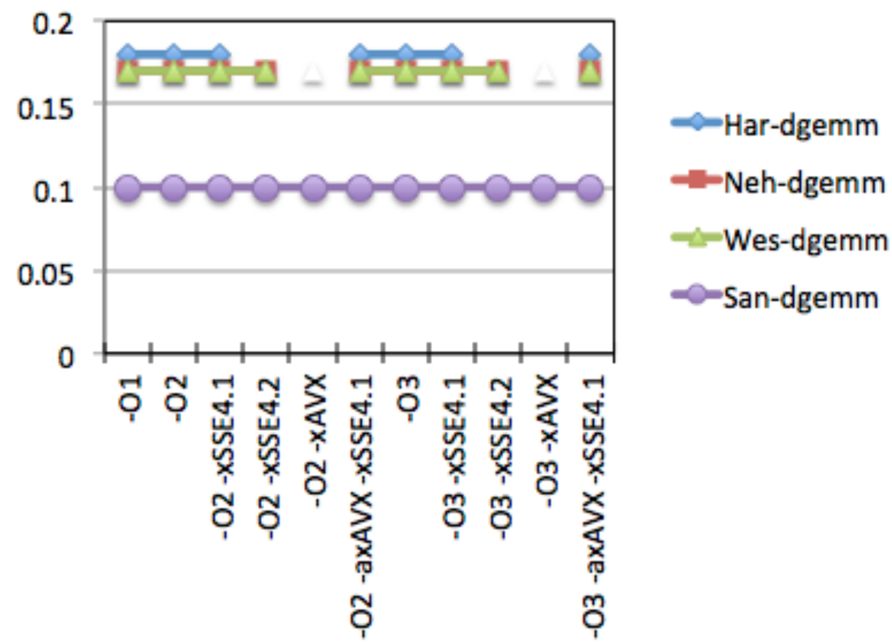
Fatal Error: This program was not built to run on the processor in your system.
The allowed processors are: Intel(R) processors with SSE4.2 and POPCNT instructions support.
[Note: POPCNT: Population count (count number of bits set to 1)]

Fatal Error: This program was not built to run in your system.
Please verify that both the operating system and the processor support Intel(R) AVX

Performance (in sec) of a Simple Matrix Multiply




```
program main
parameter (nmax=1000)
real (kind=8), dimension(nmax,nmax):: a, b, c
integer (kind=8) :: t0, t1, rate
a = 1.0
b = 2.0
c = 0.0
call system_clock(t0, rate)
do j = 1, nmax            !! call dgemm('n','n',nmax,nmax,nmax,alpha,a,lda,b,ldb,beta,c,ldc)
  do k = 1, nmax
    do i = 1, nmax
      c(i,j) = c(i,j) + a(i,k) * b(k,j)
    end do
  end do
end do
call system_clock(t1, rate)
print *, 'c(100, 201) = ', c(100,201)
print *, 'time = ', float(t1-t0)/rate
stop
end
```



Version 12.1.0 compiler used: comp-intel/2011.7.256

Question? Use the Webex chat facility to ask the Host

Performance of a Simple Matrix Multiply (cont'd)

- Time: O3 < O2 < O1 on all processor types
- –xSSE4.1 or –xSSE4.2 does not improve performance on all processor types (We saw this behavior for many codes)
- –xAVX does improve performance on Sandy Bridge
- –axAVX –xSSE4.1 gives good performance (as good as –xSSE4.2 or –xSSE4.1 on Har/Neh/Wes and as good as –xAVX on San) and also allows the executable to run on all processor types
- Using dgemm in MKL performs better than original code
- Performance of dgemm is controlled by how MKL was built by Intel. It is not sensitive to your choice of compiler flags

Warning: This example demonstrates an ideal case. Performance of your code may deviate from this observation. You should experiment yourself!!

Question? Use the Webex chat facility to ask the Host

Preliminary Performances (in sec) of SBU & some NPB Benchmarks

V12 with –O3	Cores Used	Wes	San (no –xAVX)	San (with –xAVX)
ENZO	240	1925	1854	1637
FUN3D (-O2)	960	1734	1438	1449
GEOS-5	1176	2096	1327	1235
OVERFLOW	480	1786	1200	1154
USM3D	480	1802	1583	1545
WRF	384	2036	1540	1499

V12 with –O3	Cores Used	Wes	San (no -xAVX)	San (with –xAVX)
BT.C	16	102.2	92.7	84.6
CG.C	16	26.9	15.8	16.8
FT.C	16	33.9	19.3	18.4

Source of data: ENZO: S. Heistand, FUN3D, OVERFLOW, USM3D: J. Djomehri, GEOS-5, WRF: S. Cheung, NPB: H. Jin.
Turbo Boost was ON.

Question? Use the Webex chat facility to ask the Host

Do I Need to Change My PBS Script?

- Change the number of nodes, ncpus and model

  ##PBS –lselect=4:ncpus=12:model=wes

  #PBS –lselect=3:ncpus=16:model=san

- Change the MPT version for MPI applications
  - Support for Mellanox InfiniBand FDR devices starts in MPT 2.05. MPT 2.06 has additional bug fixes and is recommended for cross-node jobs

    module load comp-intel/2011.7.256 mpi-sgi/mpt.2.06a67 (may change)
  - If you load mpt.2.04.10789 or earlier versions, job will not run:

    Get this message: *### Sandy Bridge nodes need mpt 2.06 or later*

- For better performances, may use mbind.x to pin processes/ threads if not all cores are used

  #PBS –lselect=3:ncpus=12:model=san

  # the following allows the 12 MPI processes on each node to be

  # evenly distributed between the two sockets

  mpiexec –np 36 mbind.x –cs –n12 –v $HOME/a.out > /nobackup/user

  #info about mbind.x: http://www.nas.nasa.gov/hecc/support/kb/entry/288

Question? Use the Webex chat facility to ask the Host

Should My Job Run on Sandy Bridge?

- Availability Consideration
  - 2 racks (2,304 cores) set aside for devel queue; 22 racks for normal, long, debug queues

    qsub –q devel@pbspl3 your_job_script ; *qsub –q long[@pbspl1] your_job_script*
  - Use *qstat –au foo [@pbspl3]* to check if there are free SAN nodes
  - Use *qstat –i –W o=+model,mission [@pbspl3]* to check resources requested for queued jobs
- Memory Consideration
  - 32 GB/node; some apps that got OOMs on Har/Neh/Wes may run on San
  - Sandy Bridge bigmem nodes (128/256 GB) may be available in the near future
- Performance and SBU Rates Consideration
  - Most applications should run faster (even without the use of –xAVX) on Sandy Bridge, but you should check this for your own application
  - Is it cheaper to run on Sandy Bridge than others?

Check if # of San nodes used x san wall_time x 1.65 < # of Wes nodes used x wes wall_time x 1

	San	Wes	Neh	Har
Cores/node	16	12	8	8
SBU Rate	1.65* (may change)	1	0.8	0.45

*Turbo Boost OFF

Question? Use the Webex chat facility to ask the Host

Summary

- The Sandy Bridge processor is significantly different from Har/Neh/Wes
- Recompilation of your source code with v. 12 compiler is recommended.
  (use –axAVX –xSSE4.1 if you want an executable that works on all)
- Use MKL libraries (-mkl) when possible
- Your code may or may not benefit from the AVX technology
- Most codes should benefit from other improvements in Sandy Bridge
  (larger L3, higher memory bandwidth, faster interconnect)
- Checking correctness is important
- /nobackup file systems are accessible from Sandy Bridge nodes
- Minimal modification of your PBS script is required for Sandy Bridge
- Devel (pbspl3), normal, debug, long (pbspl1) queues are available now
- Check performance and SBU usage to see if you should run on
  Sandy Bridge (*acct_query –pall –call –ujsmith –olow* for jobs finished today*)*

There may be IB or Lustre issues, but SGI/NAS continue to work on stabilizing them.
Contact NAS Help Desk if you need further assistance.

http://www.nas.nasa.gov/hecc/support/kb/Preparing-to-Run-on-Pleiades-Sandy-Bridge-Nodes_322.html